

Ch 1 Organisations & Data Management

Creating an RDMS structure, p 26

Table Normalisation: What is table normalisation? (pg30)

Table normalisation is the process of ensuring that a database conforms to a set of normal forms.

What is its primary purpose?

Its primary purpose is to remove redundancies that create threats to data integrity such as update anomalies.

There are six 'normal forms'. We need to apply the first 3 normal forms:

First normal form, (1NF)

Explain the nature of this rule.

This rule states that there must be no repeating groups in the table. This means that no single row (record) contains more than one value in a field, nor will there be more than one column with the same kind of value.

Second normal form, (2NF), p 31

Describe the nature of the 2NF.

This is where a table is in the First Normal Form and any column (field) in that table that is not part of the actual primary key must be wholly dependent on the concatenated primary key. In other words, in situations where you have more than one primary key field in a table, each non-key field must be fully dependent on both keys, not just dependent on one of the keys.

To determine if tables comply with the second normal form, look at each non-primary key field and determine if the field is wholly related to each both the primary keys. If the field can exist independently of any of the primary keys (which means it is not wholly reliant on all the keys) then the table is not in the second normal form.

Third normal form, (3NF), p 34

Describe the nature of the 3NF.

This is where a table is in the second normal form and any column that is not part of the primary key is dependant only in the on the primary key and no other column. Therefore, to be in the third normal form , every field in a table must relate directly to the primary key.

Below is a further elaboration on normalisation.

Three problems you can have with databases and why they are problems.

1. Too many "things" in one field, which makes it hard to search and sort.
2. Data repeated in many rows which means if something changes such as a customer address you need to change it in lots of places.
3. Tables that contain calculated values which take up processing power and unnecessary storage space if you have a really, really large database.

0NF – has all the problems

1NF – eliminate problem 1 (separate the data)

2NF – eliminate problem 1 and 2 (make multiple tables, add primary keys and link them)

3NF – eliminate all three problems. (eliminate calculated fields, put them in queries and reports)

1NF:

- There must be no repeated columns (fields) e.g. Contact person 1, Contact person 2, Contact person 3.
- there may only be one datum in any field - i.e. no "16kg" "3 minutes 56 seconds", "Large Size \$4, Small \$2.50", "16 Fred St, Melbourne, 3000".

2NF problems ONLY arise if you use a multi-field key (e.g. using firstname & familyname to uniquely identify people in a table).

2NF problems never even arise if each table has its own dedicated key field (e.g. ID, account number).

0NF

Smith, Sally	043123456	French, English	Epilepsy
Jones, Alan	043112345	Greek, Architecture	Asthma

1NF

Smith	Sally	043123456	French	Epilepsy
Smith	Sally	043123456	English	Epilepsy
Jones	Alan	043112345	Greek	Asthma
Jones	Alan	043112345	Architecture	Asthma

And if Alan and Sally each had multiple ailments the number of rows for these two students would rapidly increase in 1NF.

2NF requires:

- 1NF has already been achieved.
- Any non-key field in a table is dependent on ALL of the fields used as the primary key.

Below is a table with the following fields. 1NF has already been achieved.

StudentID
SubjectID
Mark
SubjectName

STUDENTID SUBJECTID Mark SubjectName

SMI0001	ENG	A+	English
SMI0001	MA	B	Maths
FRE0002	ENG	C	English

The table's key is STUDENTID and SUBJECTID (together) to uniquely identify each record in the table.

The (non-key) MARK field is dependent on both STUDENTID and SUBJECTID - i.e. to find out what a mark refers to, you need to know both the student and subject.

However, the (non-key) SUBJECTNAME field is dependent only the SUBJECTID - i.e. to find out what a subject name refers to, you only need the SUBJECTID. You don't need the STUDENTID.

So the (non-key) SUBJECTNAME field is dependent on **part of** the key (SUBJECTID) but not the **whole** key (STUDENTID+SUBJECTID).

So it fails 2NF.

To fix the problem, the table must be broken into two :

- MARKS_TABLE with STUDENTID+SUBJECTID as its key. It also contains the MARK non-key field.
- SUBJECTS_TABLE with SUBJECTID (primary key) and non-key SUBJECTNAME.

You then create a relationship between the MARKS table and the SUBJECTS table using their primary keys as the related fields.

Now, in the MARKS table, a Mark is dependent upon the entire key in its table (STUDENTID+SUBJECTID).

In the SUBJECTS table, a subject name is dependent on the entire key in its table (SUBJECTID).

This achieves 2NF.

To achieve 3NF:

- You must have already achieved 1NF and 2NF.

- No non-key field may be dependent on another non-key field.

Another way of saying it is that every non-key field in a table must give some information about the primary key rather than any other key in the table. Any field that does not contribute to the description of the primary key must be removed from the table.

For example... take a table. StudentID+SubjectID together are the primary key.

Honours is a Boolean field that is True if Mark is A or above, and False otherwise.

StudentID	SubjectID	Mark	Honours
ABC0001	S01	A	True
ABC0001	S02	A+	True
DEF0002	S01	B	False

The Honours field is dependent on the Mark field (i.e. to find the meaning of the Honours field, you need to refer to the Mark field) - but the Mark field is not the table's primary key. i.e. The Honours field describes the mark, not the student +subject.

So, a non-key field (Honours) is dependent on another non-key field (Mark). So it fails 3NF.

To fix it, do the same as we did before to achieve 2NF... break the offending field away into its own table with its own primary key (Mark) and non-key field (Honours) and relate the new table to the existing one using Mark as the link field.

This 3NF scenario looks VERY much like the 2NF before, doesn't it?

The only difference is that 2NF needed a non-key field to relate to the entire set of fields acting as the primary key.

In 3NF, it's actually simpler - a non-key field must not be dependent on another non-key field. In both cases, the fix is the same: table splitting.